

# C# Avançado

## Manipulando Documentos XML



---

---

---

---


---

---

---

---

Tópicos Abordados



- O formato XML
- API DOM
- LINQ para XML
  - Principais classes
  - Criando um documento XML
  - Lendo e gravando documentos XML
  - Manipulando dados do XML
- A classe *XmlReader*

---

---

---

---


---

---

---

---

O Formato XML



- Extensible Markup Language
- Padronizado pelo W3C
  - World Wide Web Consortium
- É um formato para representação de dados
  - Hierárquico
  - Padrão aberto e portátil
- Muito usado atualmente
  - Integrações de dados entre sistemas
  - Web services
  - Arquivos de configuração
  - Representação dos mais diversos tipos de informações

---

---

---

---

---

---

---

---

## A API DOM



- **D**ocument **O**bject **M**odel
- API especificada pelo W3C
- Permite manipular a estrutura do XML
- Esta API cria uma árvore em memória que representa a estrutura do documento XML
  - É preciso ficar atento ao consumo de memória
- Diversas linguagens de programação suportam o uso de DOM
  - No C#, o DOM é suportado através do namespace *System.Xml*
  - Atualmente, o LINQ para XML é uma alternativa mais interessante

---

---

---

---

---

---

---

## LINQ para XML



- O LINQ para XML está disponível através do namespace *System.Xml.Linq*
- É possível gerar documentos XML de forma mais simples do que usando DOM
- A extração de dados do XML pode ser feita através do uso de LINQ

---

---

---

---

---

---

---

## Principais Classes de *System.Xml.Linq*



Classe	O que representa
<i>XDocument</i>	Documento XML
<i>XElement</i>	Elemento (tag) dentro do XML
<i>XAttribute</i>	Atributo de um elemento
<i>XDeclaration</i>	Declaração no início do XML
<i>XComment</i>	Comentário
<i>XCDATA</i>	Seção CDATA de uma tag
<i>XNamespace</i>	Namespace

---

---

---

---

---

---

---

Criando um Documento XML

```

XDocument doc = new XDocument(
    new XElement("linguagens",
        new XElement("linguagem", "C#"),
        new XElement("linguagem", "Java"),
        new XElement("linguagem", "C++")
    )
);

```

A indentação ajuda a entender a estrutura do XML

```

<?xml version="1.0" encoding="utf-8"?>
<linguagens>
  <linguagem>C#</linguagem>
  <linguagem>Java</linguagem>
  <linguagem>C++</linguagem>
</linguagens>

```

---

---

---

---

---

---

---

---

Criando um Documento XML

```

XDocument doc = new XDocument(
    new XElement("linguagens",
        new XElement("linguagem", new XAttribute("ano", 2001), "C#"),
        new XElement("linguagem",
            new XElement("nome", "Java"),
            new XElement("autor", "James Gosling")
        ),
        new XElement("linguagem",
            new XElement("nome", "C++"),
            new XElement("descricao", new XCData("Linguagem orientada a objetos"))
        )
    )
);

```

```

<?xml version="1.0" encoding="utf-8"?>
<linguagens>
  <linguagem ano="2001">C#</linguagem>
  <linguagem>
    <nome>Java</nome>
    <autor>James Gosling</autor>
  </linguagem>
  <linguagem>
    <nome>C++</nome>
    <descricao><![CDATA[Linguagem orientada a objetos]]></descricao>
  </linguagem>
</linguagens>

```

---

---

---

---

---

---

---

---

Criando um Documento XML

```

XDocument doc = new XDocument(
    new XDeclaration("1.0", "iso-8859-1", "yes"),
    new XComment("Linguagens de programação"),
    new XElement("linguagens",
        new XElement("linguagem", "C#"),
        new XElement("linguagem", "Java"),
        new XElement("linguagem", "C++")
    )
);

```

```

<?xml version="1.0" encoding="iso-8859-1" standalone="yes"?>
<!--Linguagens de programação-->
<linguagens>
  <linguagem>C#</linguagem>
  <linguagem>Java</linguagem>
  <linguagem>C++</linguagem>
</linguagens>

```

O uso de *XDocument* pode ser omitido se uma declaração inicial não for necessária

---

---

---

---

---

---

---

---

## Gerando XML a Partir de Coleções



```
List<Linguagem> linguagens = new List<Linguagem>
{
    new Linguagem { Nome = "C#", Ano = 2001 },
    new Linguagem { Nome = "Java", Ano = 1995 },
    new Linguagem { Nome = "C++", Ano = 1985 }
};

XElement doc = new XElement("linguagens",
    from l in linguagens
    select new XElement("linguagem", new XAttribute("ano", l.Ano),
        new XElement("nome", l.Nome)
    )
);

<?xml version="1.0" encoding="utf-8"?>
<linguagens>
  <linguagem ano="2001">
    <nome>C#</nome>
  </linguagem>
  <linguagem ano="1995">
    <nome>Java</nome>
  </linguagem>
  <linguagem ano="1985">
    <nome>C++</nome>
  </linguagem>
</linguagens>
```

---

---

---

---

---

---

---

---

## Lendo e Gravando Documentos XML



- As classes *XDocument* e *XElement* possuem os métodos *Load()*, *Parse()* e *Save()*

```
doc.Save(@"C:\Temp\arquivo.xml");

XElement e = XElement.Load(@"C:\Temp\arquivo.xml");

StringWriter w = new StringWriter();
doc.Save(w);

string s = ...;
XElement e = XElement.Parse(s);
```

---

---

---

---

---

---

---

---

## Inserindo Elementos



- A inserção de elementos é feita através dos métodos *Add()* e *AddFirst()* de *XElement*
  - Add()*: Adiciona no final
  - AddFirst()*: Adiciona o início

```
XElement doc = new XElement("linguagens",
    new XElement("linguagem", "C#"));

XElement java = new XElement("linguagem", "Java");
doc.Add(java);

<?xml version="1.0" encoding="utf-8"?>
<linguagens>
  <linguagem>C#</linguagem>
  <linguagem>Java</linguagem>
</linguagens>
```

---

---

---

---

---

---

---

---

## Removendo Elementos



- A remoção é feita através do método *Remove()* de *XElement*
  - Pode ser usado para remover um elemento ou um fragmento completo do XML (árvore)

```
XElement doc = new XElement("linguagens",
    new XElement("linguagem", "C#"),
    new XElement("linguagem", "Java"),
    new XElement("linguagem", "C++")
);
doc.Elements("linguagem").Remove();

<?xml version="1.0" encoding="utf-8"?>
<linguagens />
```

---

---

---

---

---

---

---

## Acessando Elementos



- LINQ para XML possui alguns métodos para acessar elementos do XML
  - Definidos em *Extensions* como extension methods de *IEnumerable<T>*

Método	O que retorna
<i>Attributes()</i>	Atributos do elemento
<i>Attribute()</i>	Atributo do elemento com determinado nome
<i>Descendants()</i>	Elementos dentro de um elemento (descendentes em qualquer nível)
<i>Elements()</i>	Elementos dentro de um elemento (descendentes diretos)
<i>Element()</i>	Primeiro elemento descendente
<i>Ancestors()</i>	Pais de um elemento

---

---

---

---

---

---

---

## Acessando Elementos



- Os métodos da tabela anterior podem ser usados em expressões LINQ para extrair dados do XML

```
<?xml version="1.0" encoding="utf-8"?>
<linguagens>
  <linguagem ano="2001">
    <nome>C#</nome>
  </linguagem>
  <linguagem ano="1995">
    <nome>Java</nome>
  </linguagem>
  <linguagem ano="1985">
    <nome>C++</nome>
  </linguagem>
</linguagens>
```

```
var r = from e in doc.Elements("linguagem")
        where (int)e.Attribute("ano") > 1990
        select new
        {
            Nome = e.Element("nome").Value,
            Ano = e.Attribute("ano").Value
        };

```

Linguagem cujo ano é maior que 1990

---

---

---

---

---

---

---

## A Classe *XmlReader*



- Existem situações onde a geração da árvore do XML feita pelo DOM não é necessária ou não é viável
  - Documentos XML muito extensos
- Uma alternativa é usar a classe *XmlReader*
  - Considerado um *pull parser*
  - O XML pode ser processado enquanto é lido
  - Não existe a criação da árvore em memória
  - A navegação no XML não é tão flexível quanto no DOM
- A classe faz parte do namespace *System.Xml*

---

---

---

---

---

---

---

## Exemplo de Uso do *XmlReader*



```
<?xml version="1.0" encoding="utf-8"?>
<linguagens>
  <linguagem ano="2001">
    <nome>C#</nome>
  </linguagem>
  <linguagem ano="1995">
    <nome>Java</nome>
  </linguagem>
  <linguagem ano="1985">
    <nome>C++</nome>
  </linguagem>
</linguagens>
```

```
XmlReader reader = XmlReader.Create(@"C:\arquivo.xml");
while (reader.Read())
{
    if (reader.NodeType == XmlNodeType.Element && reader.Name == "linguagem")
    {
        int ano = int.Parse(reader.GetAttribute("ano"));
    }
    else if (reader.NodeType == XmlNodeType.Element && reader.Name == "nome")
    {
        String nome = reader.ReadElementContentAsString();
    }
}
```

---

---

---

---

---

---

---



---

---

---

---

---

---

---